# Projeto 1
# Introdução à Inteligência Artificial

**Luana Cruz Silva, 20/2033543**
**Lucas de Oliveira Silva, 200022857**

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC - Organização e Arquitetura de Computadores

***Resumo.*** *Este projeto visa a construção e apresentação de um sistema de recomendação de filmes, usando dois diferentes algoritmos de classificação: 1) Naive Bayes; 2) K-nn*

## 1. Repositório

`https://github.com/luacrz/Proj1-IIA-2023-2.git`

## 2. O que foi feito

O projeto tem como objetivo desenvolver dois sistemas de recomendação de filmes distintos, Modelo Naive Bayes e Modelo K-nn, com base no tutorial ”Beginner Tutorial: Recommender Systems in Python”, disponível em [DataCamp](https://www.datacamp.com/tutorial/recommender-systems-python).
Utilizaremos um conjunto de dados de tamanho reduzido disponível em [Kaggle](https://www.kaggle.com/rounakbanik/the-movies-dataset/data) para simplificar o processo. Para facilitar a compreensão de cada algoritmo, focaremos em duas métricas essenciais: gênero e classificação. Estas métricas são extraídas dos conjuntos de dados ’movies-metadata.csv’ e ’ratings-small.csv’. Nas próximas seções, detalharemos cada etapa desse processo.

## 3. Modelo Naive Bayes

### 3.1. Carregando e Preparando os Dados

Carregamos os conjuntos de dados ’movies-metadata.csv’ e ’ratings-small.csv’. Em seguida, calculamos a média das classificações de usuários por filme e filtramos os filmes com pelo menos 20 avaliações. Também fizemos algumas conversões de tipos de dados e combinamos os dados dos filmes com as médias das avaliações.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import numpy as np
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.naive_bayes import MultinomialNB

# Carregue o conjunto de dados
df = pd.read_csv('movies_metadata.csv')
```

```
ratings_df = pd.read_csv('ratings_small.csv')

# ...
```

### 3.2. Engenharia de Recursos - Atribuindo Rótulos com Base nos Gêneros

Criamos uma função para atribuir rótulos com base nos gêneros, onde selecionamos os gêneros alvo ('Action', 'Adventure', 'Animation', 'Comedy', 'Drama') e os aplicamos aos filmes. Em seguida, convertemos esses rótulos em uma matriz binária.

```
# Função para atribuir rótulos com base nos gêneros
def assign_genre_labels(genres):
    target_genres = ['Action', 'Adventure',
    'Animation', 'Comedy', 'Drama']

    genre_list = eval(genres)  # Converte a string
    de gêneros em uma lista de dicionários

    # Cria uma lista com os gêneros presentes no filme
    labels = [genre['name'] for genre in genre_list if genre['name'] in

    return labels

# Aplicar a função para obter as labels em formato de lista
df['label'] = df['genres'].apply(assign_genre_labels)

# Converter a lista de listas em uma matriz binária
mlb = MultiLabelBinarizer()
X = mlb.fit_transform(df['label'])
# ...
```

### 3.3. Dividindo os Dados e Treinando o Modelo

Dividimos os dados em conjuntos de treinamento e teste. Treinamos um modelo Naive Bayes Multinomial utilizando os dados de treinamento e calculamos sua acurácia.

```
# Dividir os Dados em Conjunto de Treinamento e Teste
X_train, X_test, y_train, y_test = train_test_split(X_combined,
df['label'], test_size=0.2, random_state=42)

# Treinar o modelo Naive Bayes
naive_bayes_model = MultinomialNB()
naive_bayes_model.fit(X_train, y_train)
# ...
```

### 3.4. Fazendo Previsões e Recomendações de Filmes

Fazemos uma previsão de gênero com base em uma lista de gêneros do exemplo e, em seguida, recomendamos filmes com base no gênero previsto. Também realizamos previsões no conjunto de teste.

```
# Fazer uma previsão
sample_genre_vector = mlb.transform([['Action', 'Adventure',
'Animation']])  # Gêneros do exemplo

sample_genre_vector_with_rating =
np.append(sample_genre_vector, [[3.5]], axis=1)  # Adicionar a
avaliação média

predicted_genre =
naive_bayes_model.predict(sample_genre_vector_with_rating)
# ...
```

### 3.5. Calculando Métricas de Avaliação

Calculamos métricas de avaliação, incluindo precisão, revocação e F1 para cada gênero, bem como a acurácia geral do modelo Naive Bayes. Essas métricas são importantes para avaliar o desempenho do modelo. Também foi feito a curva ROC de cada gênero.

```
# Calcular precisão, revocação, F1 e suporte para cada gênero
precision, recall, f1, support =
precision_recall_fscore_support(y_test, y_pred, average=None,
labels=genre_keywords)

# Calcular a acurácia
accuracy = accuracy_score(y_test, y_pred)

# ...


# Calcular a curva ROC para cada classe (one-vs-rest)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(len(genre_keywords)):
    y_test_binary = y_test.apply(lambda x: 1 if x == genre_keywords[i]
    y_score = naive_bayes_model.predict_proba(X_test)[:, i]
    fpr[i], tpr[i], _ = roc_curve(y_test_binary, y_score)
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plotar a curva ROC para cada classe
plt.figure()
```

```
colors = ['blue', 'red', 'green', 'purple', 'orange']
for i in range(len(genre_keywords)):
    plt.plot(fpr[i], tpr[i], color=colors[i], lw=2, label=f'Curva ROC
    para {genre_keywords[i]} (área = {roc_auc[i]:.2f})')
# ...
```

## 4. Modelo K-nn

### 4.1. Carregando e Preparando os Dados

Repetimos o processo de carregamento e preparação de dados, como no Naive Bayes.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import numpy as np
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.neighbors import KNeighborsClassifier

# Carregue o conjunto de dados
df = pd.read_csv('movies_metadata.csv')
ratings_df = pd.read_csv('ratings_small.csv')

# ...
```

### 4.2. Engenharia de Recursos - Atribuindo Rótulos com Base nos Gêneros

Atribuímos rótulos com base nos gêneros, similar ao processo anterior.

```
#...
# Função para atribuir rótulos com base nos gêneros
def assign_genre_labels(genres):
    target_genres = ['Action', 'Adventure', 'Animation', 'Comedy',
    'Drama']

    genre_list = eval(genres)  # Converte a string de gêneros em uma
    lista de dicionários

    # Cria uma lista com os gêneros presentes no filme
    labels = [genre['name'] for genre in genre_list if genre['name']
    in target_genres]

    return labels
```

```
# Aplicar a função para obter as labels em formato de lista
df['label'] = df['genres'].apply(assign_genre_labels)

# Converter a lista de listas em uma matriz binária
mlb = MultiLabelBinarizer()
X = mlb.fit_transform(df['label'])
#...
```

### 4.3. Dividindo os Dados e Treinando o Modelo K-nn

Dividimos os dados em conjuntos de treinamento e teste. Treinamos um modelo K-nn com um número específico de vizinhos (neste caso, k=5) utilizando os dados de treinamento e calculamos sua acurácia.

```
#...
# Dividir os Dados em Conjunto de Treinamento e Teste
X_train, X_test, y_train, y_test = train_test_split(X_combined,
df['label'], test_size=0.2, random_state=42)

# Treinar o modelo k-NN
k = 5  # Número de vizinhos
knn_model = KNeighborsClassifier(n_neighbors=k)
knn_model.fit(X_train, y_train)
#...
```

### 4.4. Fazendo Previsões e Recomendações de Filmes

Fizemos uma previsão de gênero com base em uma lista de gêneros do exemplo e recomendamos filmes com base no gênero previsto. Também realizamos previsões no conjunto de teste.

```
# Fazer uma previsão
sample_genre_vector = mlb.transform([['Action', 'Adventure',
'Animation']])  # Gêneros do exemplo
sample_genre_vector_with_rating = np.append(sample_genre_vector,
[[3.5]], axis=1)  # Adicionar a avaliação média

predicted_genre = knn_model.predict(sample_genre_vector_with_rating)
print(f'Gênero Previsto: {predicted_genre[0]}')
#...
```

### 4.5. Calculando Métricas de Avaliação

Calculamos métricas de avaliação, incluindo precisão, revocação e F1 para cada gênero, bem como a acurácia geral do modelo K-nn.

```
# Calcular precisão, revocação, F1 e suporte para cada gênero
precision, recall, f1, support =
```

```
precision_recall_fscore_support(y_test, y_pred, average=None,
labels=genre_keywords)

# Calcular a acurácia
accuracy = accuracy_score(y_test, y_pred)

# ...

# Calcular a curva ROC para cada classe (one-vs-rest)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(len(genre_keywords)):
    y_test_binary = y_test.apply(lambda x: 1 if x ==
    genre_keywords[i] else 0)
    y_score = knn_model.predict_proba(X_test)[:, i]
    fpr[i], tpr[i], _ = roc_curve(y_test_binary, y_score)
    roc_auc[i] = auc(fpr[i], tpr[i])

# ...
```

## 5. Saídas

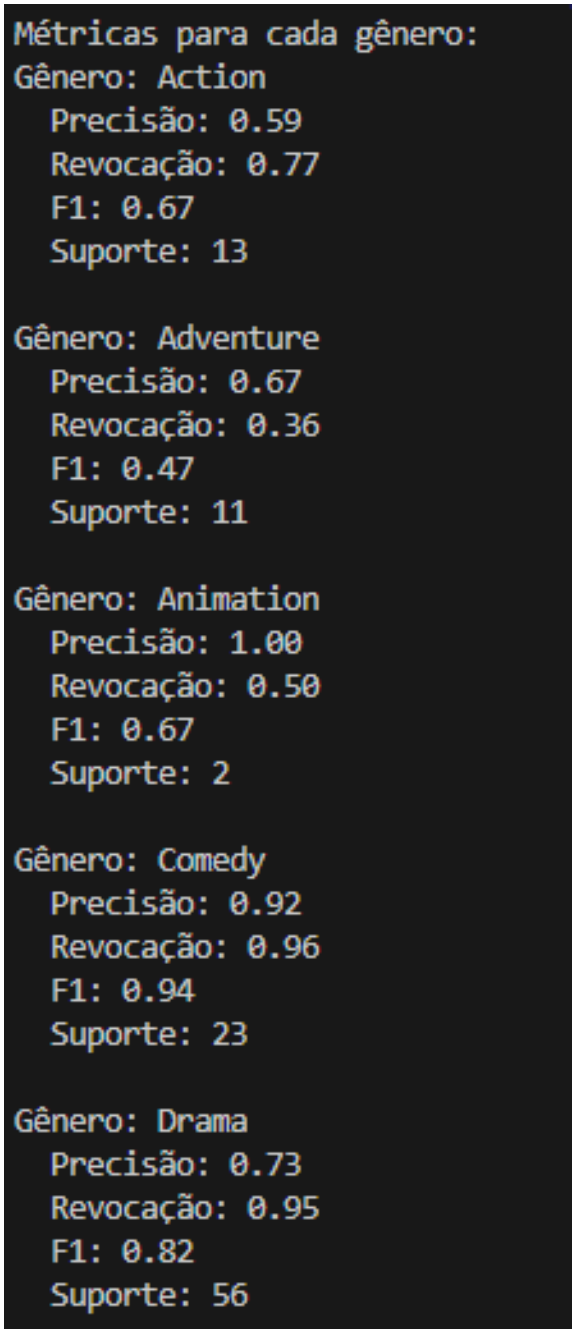


**Figura 1. Métricas Naves Bayes.**

```
Gênero Previsto: Action
Filmes Recomendados:
                          title                                              genres
0                Cutthroat Island  [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
13         Star Trek: Generations  [{'id': 878, 'name': 'Science Fiction'}, {'id'...
19  Faster, Pussycat! Kill! Kill!  [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
20          Beverly Hills Cop III  [{'id': 28, 'name': 'Action'}, {'id': 35, 'nam...
23                    Hard Target  [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
24                  Judgment Night  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
28              Romeo Is Bleeding  [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
30                   True Romance  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
31     Terminator 2: Judgment Day  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
46                    The 39 Steps  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
Métricas para cada gênero:
```

**Figura 2. Recomendação Naves Bayes.**

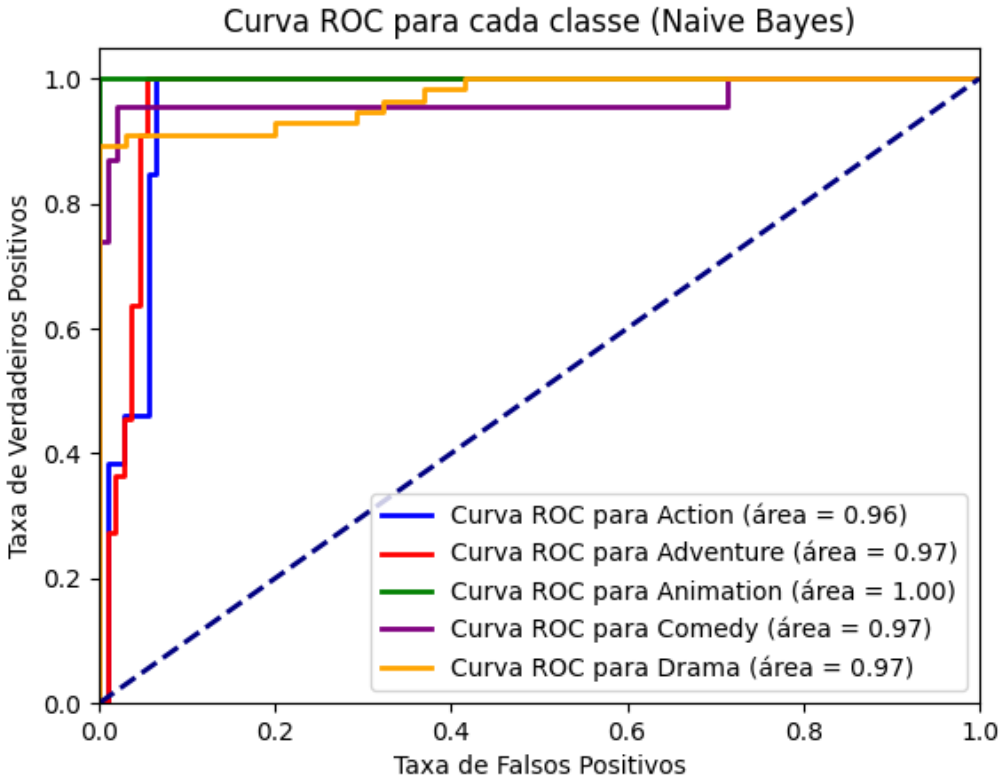


**Figura 3. Curva ROC do Naves Bayes.**

```
Gênero Previsto: Action
Filmes Recomendados:
                          title                                              genres
0               Cutthroat Island  [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
13        Star Trek: Generations  [{'id': 878, 'name': 'Science Fiction'}, {'id'...
19  Faster, Pussycat! Kill! Kill!  [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
20         Beverly Hills Cop III  [{'id': 28, 'name': 'Action'}, {'id': 35, 'nam...
23                   Hard Target  [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
24                 Judgment Night  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
28             Romeo Is Bleeding  [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...
30                 True Romance  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
31    Terminator 2: Judgment Day  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
46                 The 39 Steps  [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
Métricas para cada gênero:
```

**Figura 4. Recomendação k-NN.**

```
Métricas para cada gênero:
Gênero: Action
  Precisão: 0.71
  Revocação: 0.92
  F1: 0.80
  Suporte: 13

Gênero: Adventure
  Precisão: 0.64
  Revocação: 0.64
  F1: 0.64
  Suporte: 11

Gênero: Animation
  Precisão: 0.50
  Revocação: 0.50
  F1: 0.50
  Suporte: 2

Gênero: Comedy
  Precisão: 0.88
  Revocação: 0.91
  F1: 0.89
  Suporte: 23

Gênero: Drama
  Precisão: 0.98
  Revocação: 0.89
  F1: 0.93
  Suporte: 56
```

**Figura 5. Métricas do k-NN.**

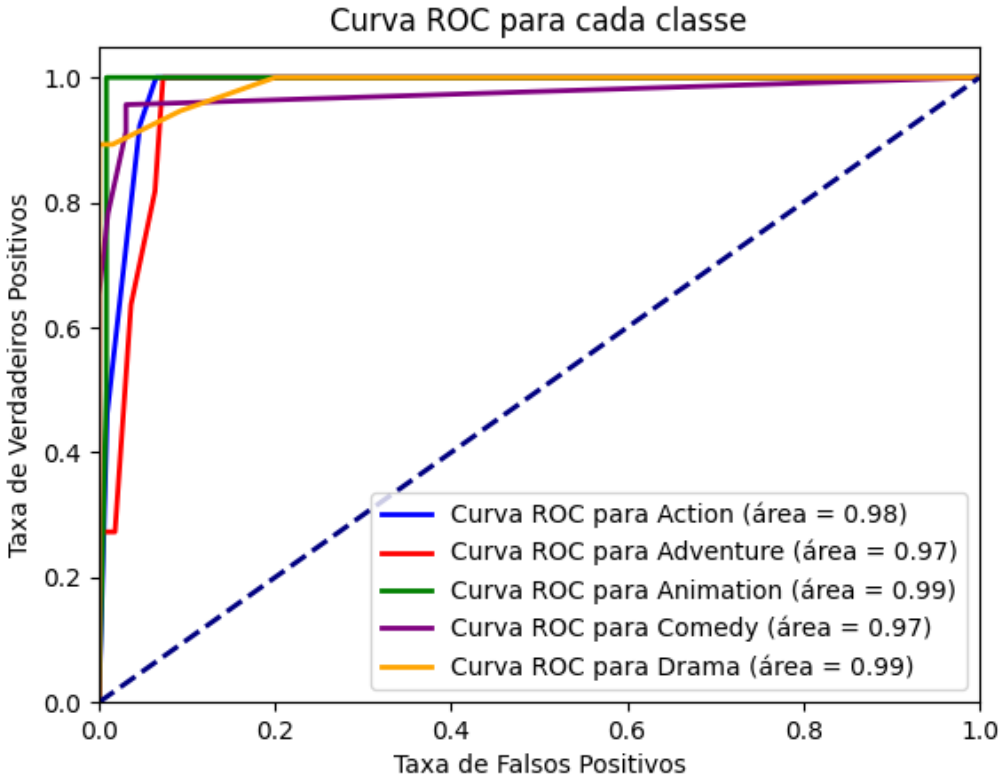


**Figura 6. Curva ROC do k-NN.**

## 6. Conclusão

Ambos os algoritmos, Naive Bayes e K-nn, foram aplicados ao conjunto de dados para a recomendação de filmes com base em gênero e avaliação. Foram calculadas métricas de avaliação para avaliar o desempenho desses modelos.

O código completo pode ser encontrado em um Jupyter Notebook, conforme mencionado na solicitação. Mas recomendamos que seja executado os arquivos .py do repositório 1.

## Referências